UNIT IV MESSAGE AUTHENTICATION AND INTEGRITY

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – SHA –Digital signature and authentication protocols – DSS- Entity Authentication: Biometrics, Passwords, Challenge Response protocols- Authentication applications - Kerberos, X.509

AUTHENTICATION REQUIREMENT

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis**: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination.

MESSAGE AUTHENTICATION FUNCTION

Message Authentication Function can be grouped into three classes.

• **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator

• **Message encryption:** The cipher text of the entire message serves as its authenticator

• **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

HASH FUNCTION

Hash function accepts a variable size message M as input and produces a fixed size output, referred to as hash code H(M). The hash code does not use any key. The hash code is also referred to as message digest or hash value. The hash code is a function of all the bits of the message and provides an error detection capability. Figure illustrates a variety of ways in which a hash code can be used to provide message authentication.

a) The message plus concatenated hash code is encrypted using symmetric encryption. The encryption is applied to the entire message plus hash code, confidentiality is also provided.

b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c) Only the hash code is encrypted, using public-key encryption and using the sender's private key. This provides authentication.

d) If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

e) The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recomputed the hash value to verify.

f) Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

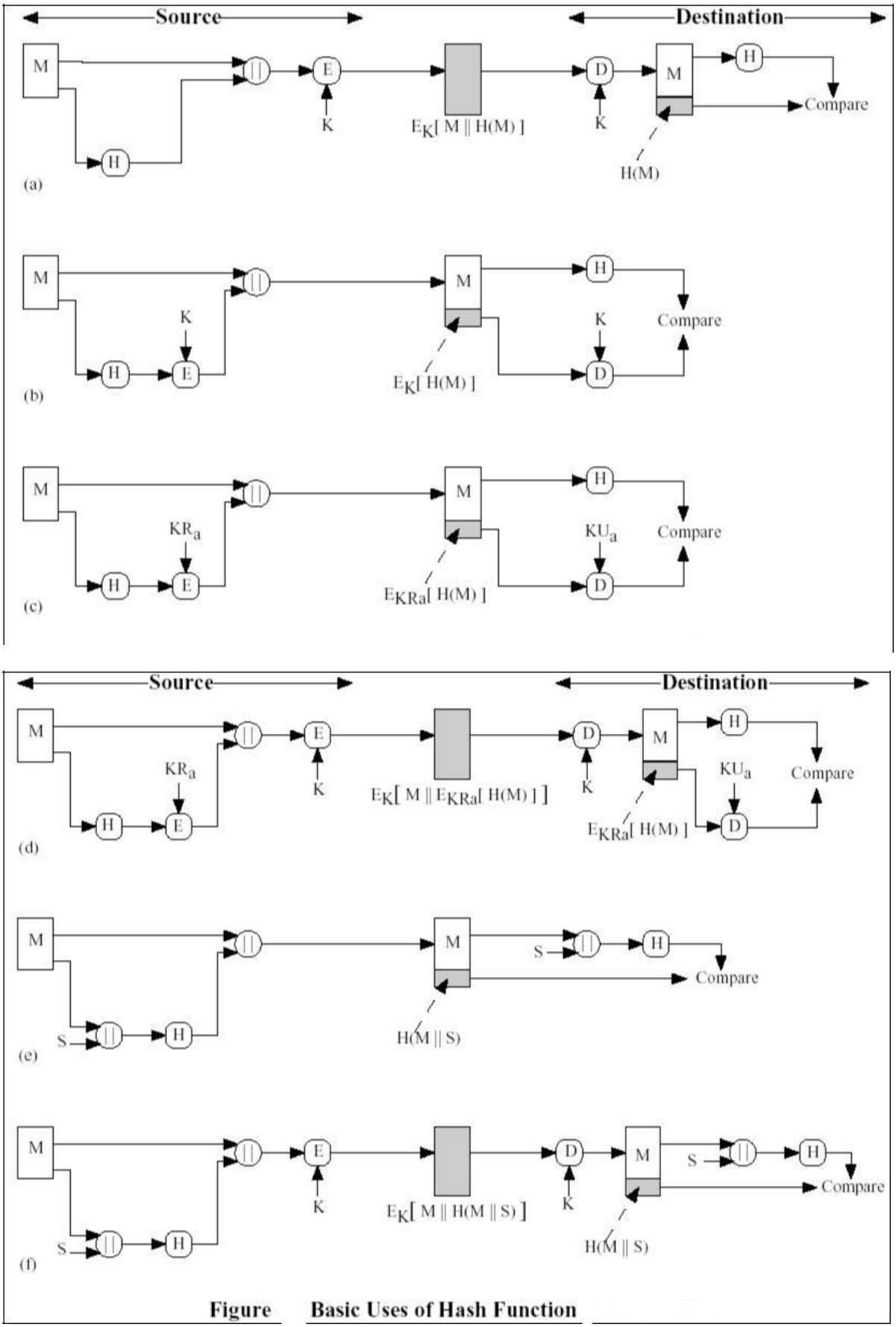


Figure **Basic Uses of Hash Function**

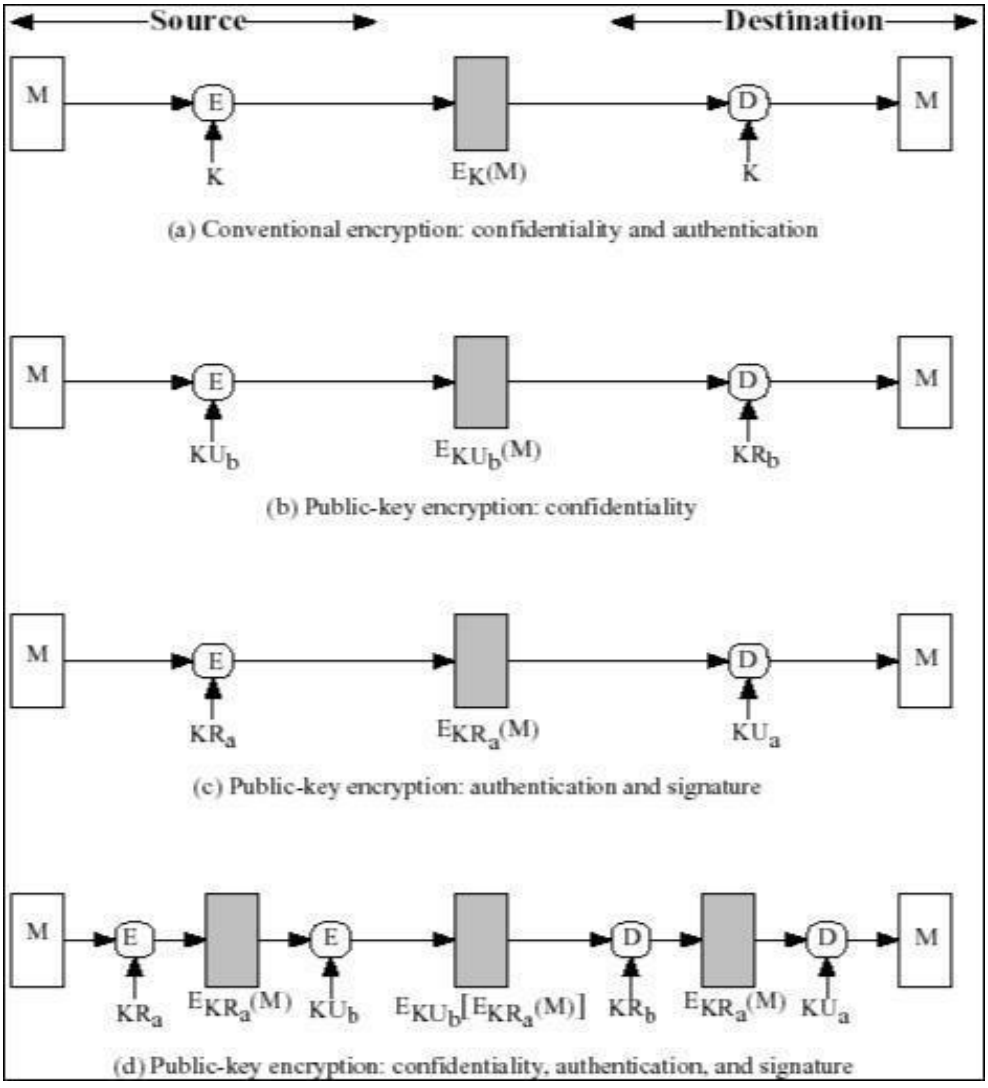
MESSAGE ENCRYPTION

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

Symmetric Encryption

Conventional encryption provides *authentication* as well as *confidentiality*. Message *M* transmitted from source A to destination B is encrypted using a secret key *K* shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.

Given a decryption function D and a secret key *K*, the destination will accept *any* input *X* and produce output $Y = D(K, X)$. If *X* is the ciphertext of a legitimate message *M* produced by the corresponding encryption function, then *Y* is some plaintext message *M*. Otherwise, *Y* will likely be a meaningless sequence of bits.



Public-Key Encryption

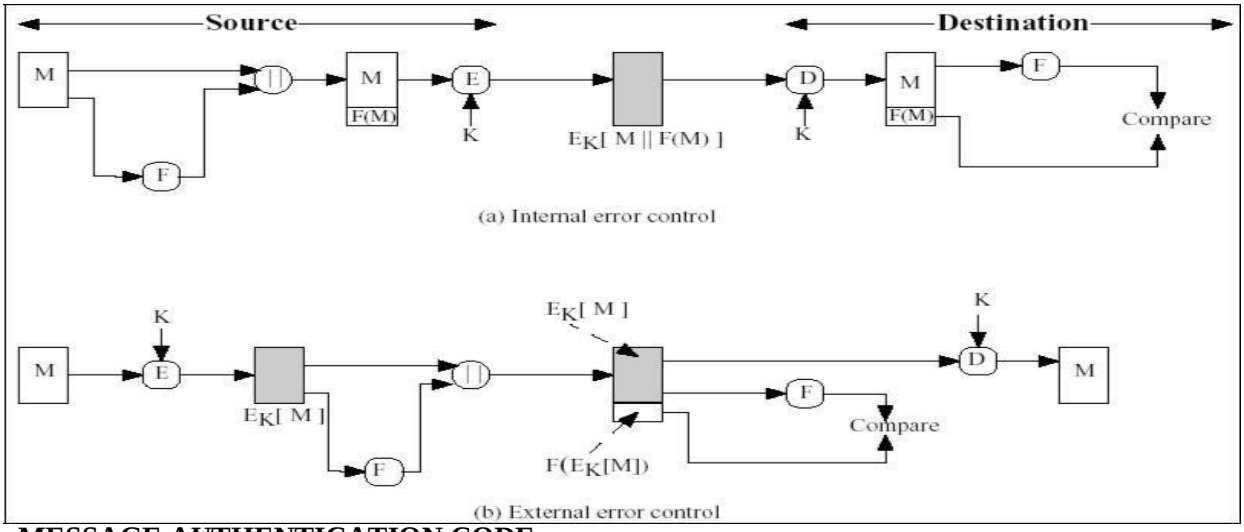
- The straightforward use of public-key encryption provides confidentiality but not authentication. The source (A) uses the public key P_{Ub} of the destination (B) to encrypt M. Because only B has the corresponding private key P_{Rb}, only B can decrypt the message.
- To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt

- To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality.

Ways of Providing *Structure*

- Append an error-detecting code (frame check sequence (FCS)) to each message
- A prepares a plaintext message M and provides this as input to a function F that produces an FCS.
- The FCS is appended to M and the entire block is then encrypted.
- At the destination, B decrypts the incoming block and treats the results as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS.

If the calculated FCS is equal to the incoming FCS, then the message is considered authentic



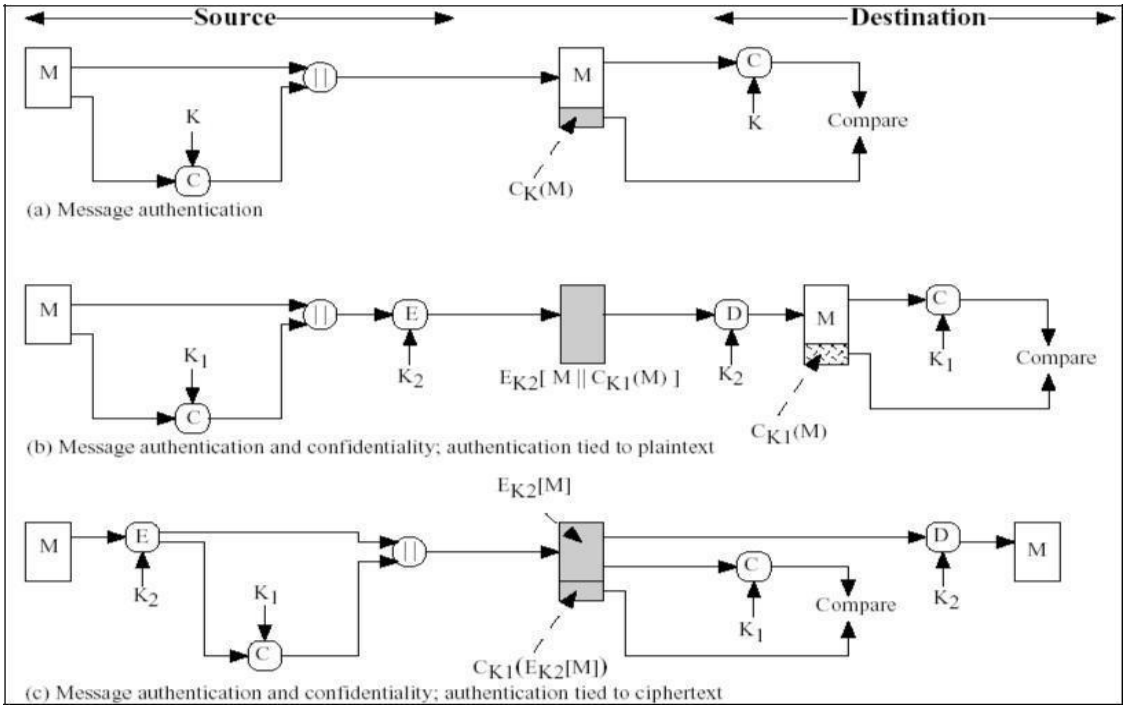
MESSAGE AUTHENTICATION CODE

Uses a shared secret key to generate a fixed-size block of data (known as a cryptographic checksum or MAC) that is appended to the message.

$$MAC = C_K(M)$$

Where *M* is a variable-length message, *K* is a secret key shared only by sender and receiver, and $C_K(M)$ is the fixed-length authenticator. The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption



- In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted.
- In the second case, the message is encrypted first. Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block

SECURE HASH ALGORITHM (SHA)

The most widely used hash function has been the Secure Hash Algorithm (SHA). SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.SHA is based on the hash function MD4, and its design closely models MD4. Three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively. Collectively, these hash algorithms are known as SHA-2.

SHA-512 Logic

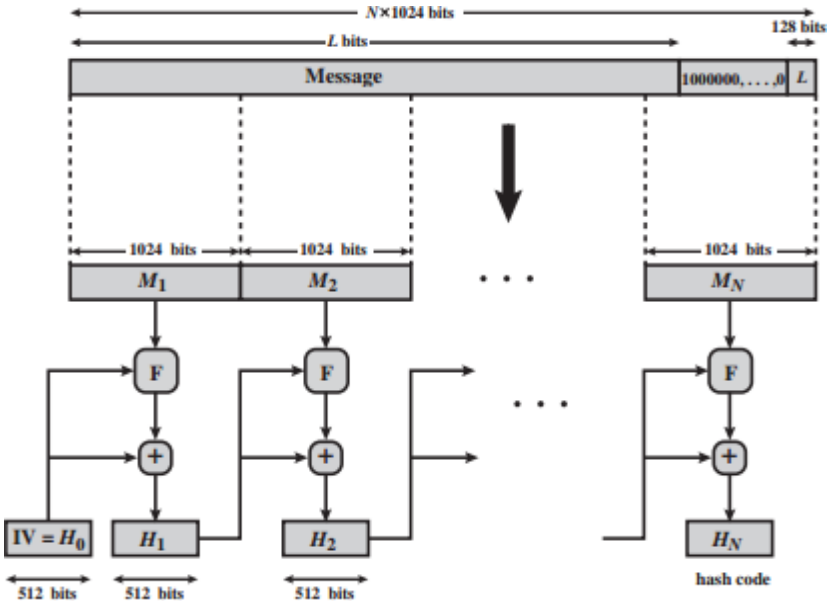
Message Digest Generation Using SHA-512

The algorithm takes as input a message with a maximum length of less than bits 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. The processing consists of the following steps.

Step 1 Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024 of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

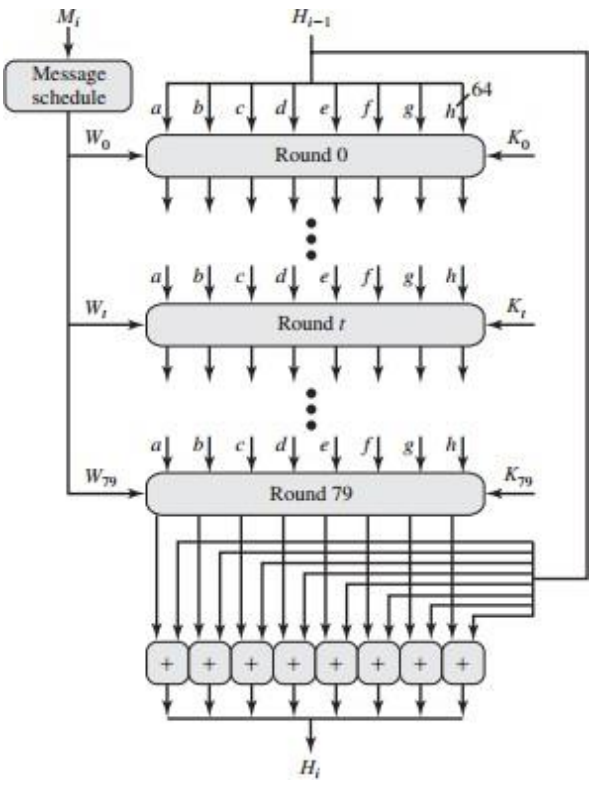
Step 2 Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. The expanded message is represented as the sequence of 1024-bit blocks *M*1, *M*2, Á , *MN* , so that the total length of the expanded message is *N* * 1024 bits

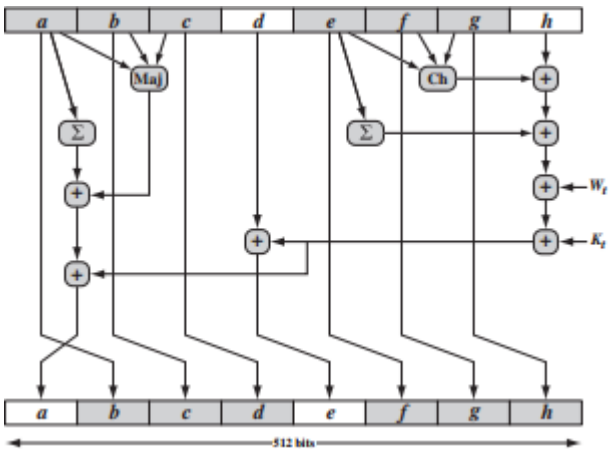


Step 3 Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).These registers are initialized to the following 64-bit integers (hexadecimal values) These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position.

Step 4 Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; Each round takes as input the 512-bit buffer value, ABCDEFGH, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, *Hi*-1).



SHA-512 Processing of a Single 1024-Bit Block



Compression Function

Step 5 Output. After all 1024-bit blocks have been processed, the output from the N th stage is the 512- bit message digest. Thus, in the first 16 steps of processing, the value of W_t is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t , with two of those values subjected to shift and rotate operations.

ELGAMAL DIGITAL SIGNATURES

Elgamal signature scheme involves the use of private key for encryption and public key for decryption

The global elements of Elgamal digital signature are prime number q and a, which is the primitive root of q.

1. Global Public key Components

q - prime no.
a – primitive root of q

2. User A signs a message M to B by computing

- Generate a random integer X_A, such that $1 < X_A < q-1$
- Compute $\mathbf{Y}_A = a^{X_A} \bmod q$
- A's Private key is X_A
- A's Public key is Y_A

To sign a message M, user A first computes the hash m=H(M), such that m is an integer in the range $0 \leq m \leq (q-1)$

3. User A generates the digital signature

- Choose a random integer K, such that $1 \leq K \leq (q-1)$ and gcd(K,q-1) = 1. That is, K is relatively prime to q-1.
- Compute, S1 = a K mod q
- Compute K -1 mod q-1
- Compute, S2 = K-1(m-xAS1) mod (q-1)
- The signature consists of a pair (S_1,S_2)
-

2. User B verifies the Signature

$V_1 = a^m \bmod q$

$V_2 = (YA)^{S1} (S_1)^{S2} \bmod q$

The signature is valid if $V_1 = V_2$.

Example I Global Element

q=19 and a=10

Alice computes the private and public key

- Alice computes her key:
  - Alice chooses Private key, X_A=16
  - Computes Public Key, $Y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash m=14
  - Alice chooses K=5 which is relatively prime to q-1=18
  - Compute $S_1 = 10^5 \bmod 19 = 3$
  - Compute $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - Compute $S_2 = 11(14-16*3) \bmod 18 = -374 \bmod 18=4$ {-374 mod 18=18-374%18}
- B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$
  - Since 16 = 16 signature is verified and valid.

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that e = H(M|| x').

To see that the verification works, observe that $x' \equiv a^y v_e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$

Hence, H(M|| x`) = H(M||x)

DIGITAL SIGNATURE STANDARD (DSS)

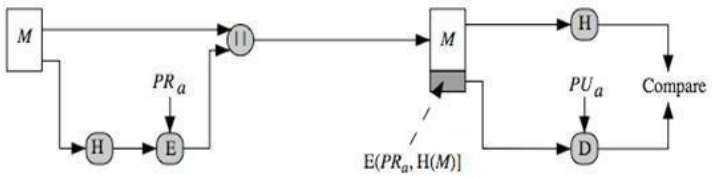
Digital Signature Standard is an US Govt approved signature scheme FIPS 186. It uses the SHA hash algorithm. It is designed by NIST & NSA in early 90's. It creates a 320 bit signature, but with 512-1024 bit security.

Two approaches of Digital signature

- RSA approach
- DSS approach

RSA APPROACH

In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.



(a) RSA Approach

M – Message

H(M) – Hash function of M C – Cipher text

$C = M \| E_{PRa}[H(M)]$

$V_1=D_{PUa}[H(M)]$

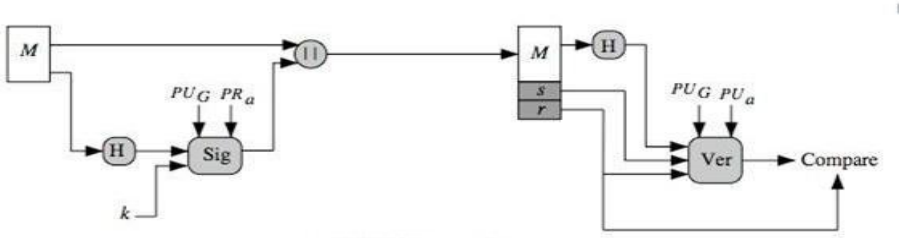
$V_2= H(M)$

If $V_1= V_2$ Then signature is verified

DSS APPROACH

DSS uses an algorithm that is designed to provide only digital signature function. Unlike RSA, it cannot be used for encryption or key exchange.

The DSS approach makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PRa) and the global public key (PUG) The result is a signature consisting of two components, labeled *s* and *r*. At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function.



(b) DSS Approach

The verification function also depends on the global public key as well as the sender's public key (PUa), which is paired with the sender's private key. The output of the verification function is a

value that is equal to the signature component if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

THE DIGITAL SIGNATURE ALGORITHM

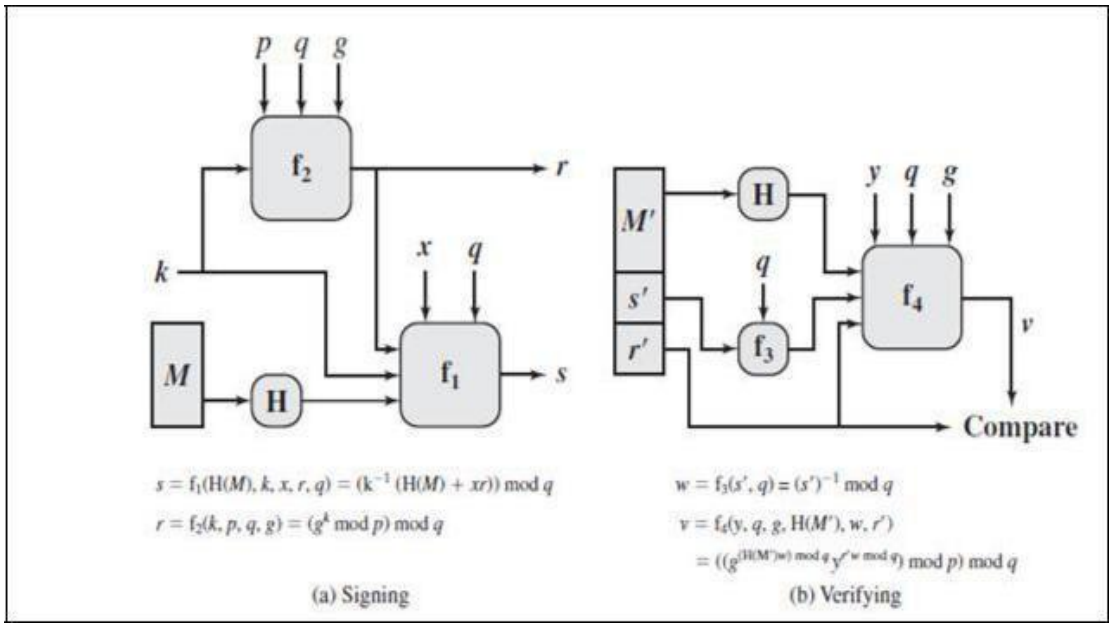
1. Global Public key Components

p- prime no. where $2^{L-1} < p < 2^{L}$ for $512 \leq L \leq 1024$ q –

prime divisor of (p-1) where 2

$g = h^{(p-1)/q} \bmod p$

where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 12\ 160$



2. User's Private key

x - random or pseudo random integer with $0 < x < q$

3. User's Public key

$y = g^{x} \bmod p$

4. User's Per Message Secret Number

k = random or pseudo random integer with $0 < k < q$

5. DSA Signature Creation

To sign a message M the sender: the sender generates a random signature key k, k<q

Computes signature pair: r =

$(g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature = (r,s)

6. DSA Signature Verification

After received M & signature (r,s)

Verify a signature, recipient computes: w = $(s')^{-1} \bmod q$

u1= [H(M')w]mod q u2= (r'w)mod q

$v = [(g^{u1} y^{u2}) \bmod p \,] \bmod q$

If v=r then signature is verified.

AUTHENTICATION PROTOCOLS

Two types of authentication protocols are

- Mutual authentication
- One way authentication

MUTUAL AUTHENTICATION

Mutual authentication protocols enable communicating parties to satisfy themselves mutually about each other's identity and exchange session keys. Problems faced by authenticated key exchange are

o Confidentiality

o Timeliness

Following are the examples of replay attack

Simple replay: The opponent simply copies a message and replays it later.

Repetition that can be logged: An opponent can replay a time stamped message within the valid time window.

- Repetition that cannot be detected: This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
- Backward replay without modification: This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

Countermeasures include

- Timestamps: Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.
- Challenge/response: Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

USING SYMMETRIC ENCRYPTION

Two-level hierarchy of symmetric key encryption can be used to provide confidentiality for the distribution of key. Here, a trusted Key Distribution Center (KDC) is used for the distribution of session key.

Needham-Schroeder Protocol for Distribution of session key

The purpose of this protocol is to distribute securely a session key Ks to A and B. Secret keys Ka and Kb are shared between A and KDC; and KDC and B respectively.

1. A→KDC: $ID_A \| ID_B \| N_1$
2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]\ ]$
3. A→B: $E_{Kb}[Ks \| ID_A]$
4. B→A: $E_{Ks}[N_2]$
5. A→B: $E_{Ks}[f(N_2)]$

This protocol is vulnerable to a replay attack. Consider that opponent X has got the old session key that was used by A and B, Now, X can impersonate A and trick B using the old key, replay step3. To overcome this attack another protocol was proposed by Denning.

Denning Protocol

In this protocol a timestamp T was used that assures A and B that the session key has only just been generated.

A→KDC: $ID_A \| ID_B$
KDC→A: E_{Ka}[Ks || ID_B ||T || E_{Kb}[$Ks||ID_{A||T}$]]
A→B: E_{Kb}[$Ks||ID_{A||T}$]
B→A: E_{Ks}[N_2]
A→B: E_{Ks}[f(N_2)]
USING PUBLIC-KEY ENCRYPTION

Protocol 1:Denning AS Protocol

Denning 81 presented the following:
A→AS: $ID_A \| ID_B$
AS→A: E_{KRas}[ID_A||KU_a||T] || E_{KRas}[ID_B||KU_b||T]
A→B: E_{KRas}[ID_A||KU_a||T] || E_{KRas}[ID_B||KU_b||T] || E_{KUb}[E_{KRas}[K_s||T]]
AS is an authentication server which provides the certificate. Protocol 2: By Woo

Another approach, proposed by Woo and Lam, makes use of nonces. The protocol consists of the following steps:

1. A →KDC:IDA||IDB
2. KDC →A:E(PRauth, [IDB||PUb])
3. A →B: E(PUb, [Na||IDA])
4. B →KDC:IDA||IDB||E(PUauth, Na)
5. KDC →B:E(PRauth, [IDA||PUa])||E(PUb, E(PRauth, [Na||Ks||IDA||IDB])
6. B →A: E(PUa, E(PRauth, [(Na||Ks||IDA||IDB)||Nb]))
7. A →B: E(Ks, Nb)

Explanation

1. In step 1, A informs the KDC of its intention to establish a secure connection with B.
2. The KDC returns to A a copy of B's public-key certificate (step 2).
3. Using B's public key, A informs B of its desire to communicate and sends a nonce N_a (step 3).
4. In step 4, B asks the KDC for A's public-key certificate and requests a session key; B includes A's nonce so that the KDC can stamp the session key with that nonce. The nonce is protected using the KDC's public key.
5. In step 5, the KDC returns to B a copy of A's public-key certificate, plus the information {N_a, K_s, ID_B}. This information basically says that K_s is a secret key generated by the KDC on behalf of B and tied to N_a; the binding of K_s and N_a will assure A that K_s is fresh. This triple is encrypted, using the KDC's private key, to allow B to verify that the triple is in fact from the KDC. It is also encrypted using B's public key, so that no other entity may use the triple in an attempt to establish a fraudulent connection with A.
6. In step 6, the triple {N_a, K_s, ID_B}, still encrypted with the KDC's private key, is relayed to A, together with a nonce N_b generated by B. All the foregoing are encrypted using A's public key. A retrieves the session key K_s and uses it to encrypt N_b and return it to B.
7. This last message assures B of A's knowledge of the session key.

One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email) have

header in clear so can be delivered by ema

- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

This protocol is compact but, as before, requires synchronization of clocks. Another approach, proposed by Woo and Lam, makes use of nonces. The protocol consists of the following steps:

1. A →KDC:IDA||IDB
2. KDC →A:E(PRauth, [IDB||PUb])
3. A →B: E(PUb, [Na||IDA])
4. B →KDC:IDA||IDB||E(PUauth, Na)
5. KDC →B:E(PRauth, [IDA||PUa])||E(PUb, E(PRauth, [Na||Ks||IDA||IDB])
6. B →A: E(PUa, E(PRauth, [(Na||Ks||IDA||IDB)||Nb]))
7. A →B: E(Ks, Nb)

KERBEROS

- Kerberos is an authentication service developed by MIT and is one of the best known and most widely implemented **trusted third party** key distribution systems.
- Provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Kerberos Requirements

Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
Reliable: Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
Transparent: The user should not be aware that authentication is taking place, beyond the requirement to enter a password.
Scalable: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.
Kerberos is a basic third-party authentication scheme.

Authentication Server (AS)

- Knows the passwords of all users and stores these in a centralized database.
- AS shares a unique secret key with each server.
- These keys have been distributed physically or in some other secure manner
- users initially negotiate with AS to identify self
- AS provides a non-corruptible authentication credential (ticket granting ticket TGT)

Ticket Granting server (TGS)

- issues tickets to users who have been authenticated to AS
- users subsequently request access to other services from TGS on basis of users TGT

Simple Authentication Dialogue

(1) C →AS: $ID_C||P_C||ID_V$
(2) AS →C: Ticket
(3) C →V : $ID_C||$Ticket

Ticket = $E(K_v, [ID_C||AD_C||ID_V])$

Where

C	= client	ID_V	= identifier of V
AS	= authentication server	P_C	= password of user on C
V	=server	AD_C	= network address of C
ID_C	= identifier of user on C	K_v	= secret encryption key shared by AS and V

Drawback of simple authentication dialogue

- The password Pc is transmitted as a simple plain text. So, there is a possibility of capturing by the attacker.

<u>More secure authentication Dialogue</u>

Table: Kerberos Version 4 Message Exchanges

(1) C →AS $ID_c||ID_{tgs}||TS_1$

(2) AS →C $E(K_c,[K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_c||AD_c||ID_{tgs}||TS_2||Lifetime_2]$

Authentication Service Exchange to obtain ticket-granting ticket

(3) C→TGS $ID_v||Ticket_{tgs}||Authenticator_c$

(4) TGS →C $E(K_{c,tgs}, [K_{c,v}||ID_v||TS_4||Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v}||ID_C||AD_C||ID_v||TS_4||Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C||AD_C||TS_3])$

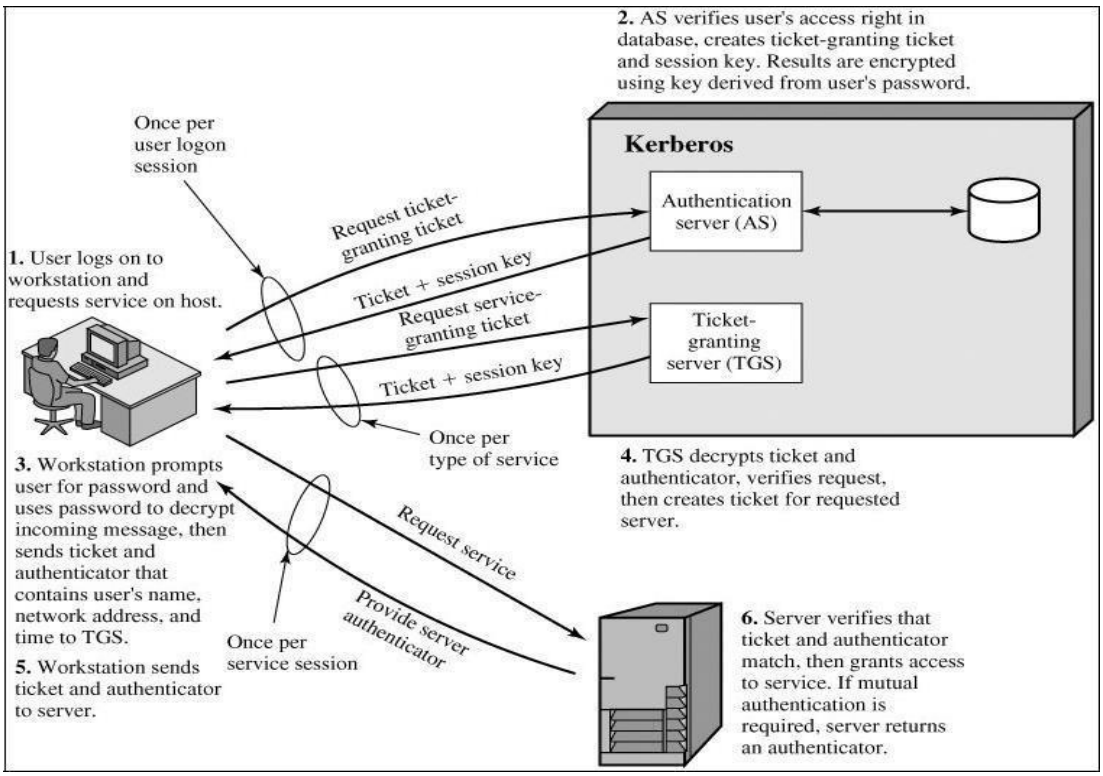
Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C →V $Ticket_v||Authenticator_c$

(6) V→C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v}||ID_c||AD_c||ID_v||TS_4||Lifetime_4])$

$Authenticator_c = E(K_{c,v},[ID_c||AD_C||TS_5])$



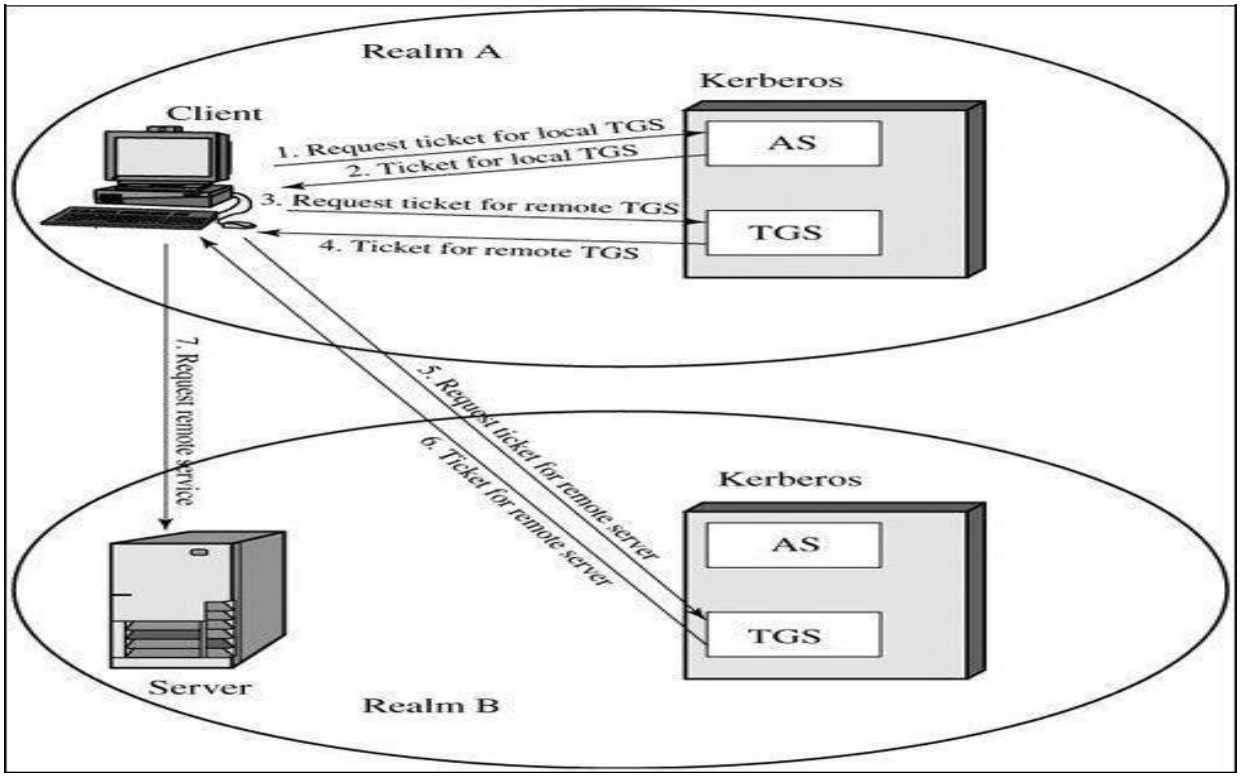
Overview of Kerberos

- Client sends a message to the AS requesting access to the TGS.
- AS responds with a message, encrypted with a key derived from the user's password (K_c) that contains the ticket.
- Encrypted message also contains a copy of the session key, $K_{c,tgs}$, where the subscripts indicate that this is a session key for C and TGS.
- Session key is inside the message encrypted with K_c, only the user's client can read it.
- Same session key is included in the ticket, which can be read only by the TGS.
- Thus, the session key has been securely delivered to both C and the TGS.
- Message (1) includes a timestamp, so that the AS knows that the message is timely.
- Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the T

Kerberos Realms

Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.
3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.



Such an environment is referred to as a **Kerberos realm**. The concept of *realm* can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database.

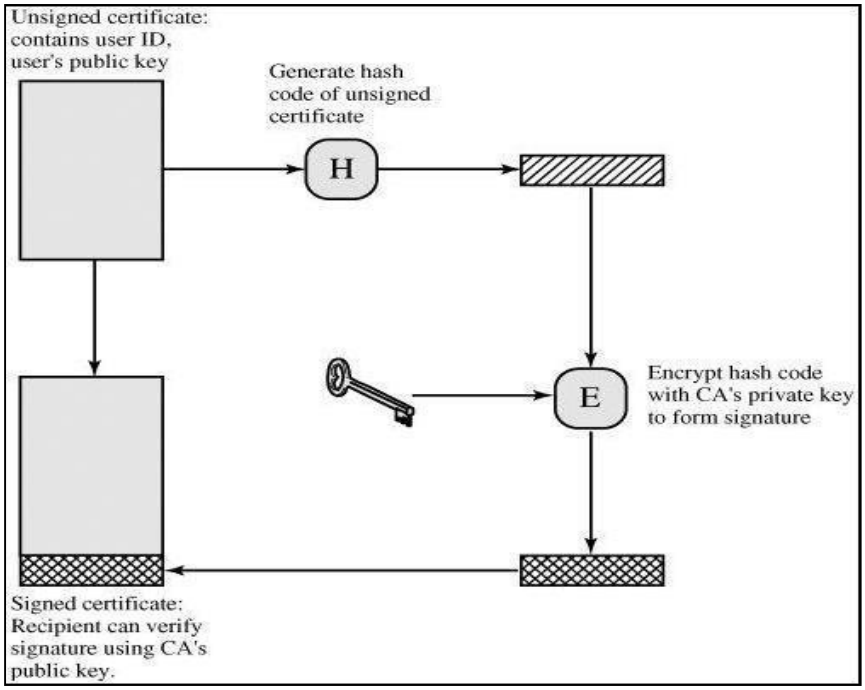
Kerberos principal, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name

A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm). The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS.

X.509 AUTHENTICATION SERVICE

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates of the type.

Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS and SET. X.509 is based on the use of public-key cryptography and digital signature algorithms. Figure illustrates the generation of public key.



Certificates

Figure shows the general format of a certificate, which includes the following elements: **Version:** Differentiates among successive versions of the certificate format; the default is version 1.

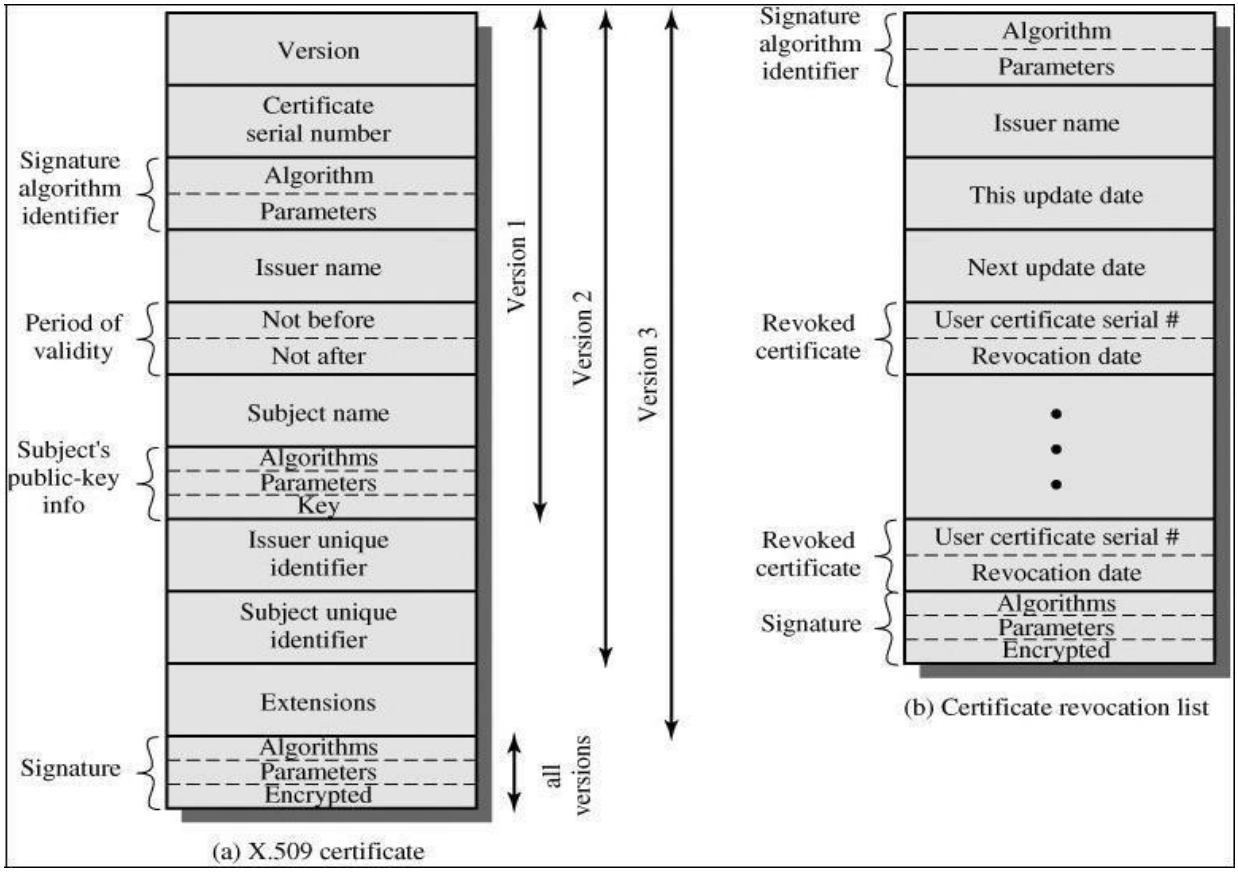
Serial number: An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.

Signature algorithm identifier: The algorithm used to sign the certificate, together with any associated parameters

Issuer name: X.500 name of the CA that created and signed this certificate.
Period of validity: Consists of two dates: the first and last on which the certificate is valid. **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
Subject's public-key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
Issuer unique identifier: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
Subject unique identifier: An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

Extensions: A set of one or more extension fields.

Signature: This field includes the signature algorithm identifier.



The standard uses the following notation to define a certificate:

$CA\langle\langle A\rangle\rangle = CA \{V, SN, AI, CA, T_A, A, Ap\}$

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

Obtaining a Certificate

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate without this being detected. Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

Certificate Revocation

Certificates have a period of validity.

- May need to revoke before expiry, eg:
  - User's private key is compromised
  - User is no longer certified by this CA
  - CA's certificate is compromised

CA maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. Each certificate revocation list (CRL) posted to the directory is signed by the issuer. When a user receives a certificate in a message, the user must determine whether the certificate has been revoked. The user could check the directory each time a certificate is received.

Entity Authentication Protocols

Entity authentication in today's IT [W11] security industry is used as one part of a security system to verify that an entity actually is who or what they claim to be prior to allowing them access to secured information or into a secured area. Entity authentication, whether cryptographic, biometric or otherwise, is a major part of today's IT security industry. As a result, there is much development in the field, leading simultaneously to both rapid advances and a pronounced need for IT security standards focusing on interoperability and the latest secure mechanisms.

The simplest example of this is when somebody fills out a username and password to log into their account on a website. Their username is a claim to an identity and their password, presumably known only to them, serves to verify their claim. While this appears simple to the user, there is quite a lot going on behind the scenes. Accommodating this variety within the IT security industry, different IT security standards have been published to address different implementations of cryptographic entity authentication.

Entity authentication protocol mainly deals with the password, biometric and challenge-response based schemes. Where password authentication is a conventional scheme and it has a 'user id' and a 'password'. User id acts like a claim and password as evidence supporting the claim. Biometric features [SARA11] like fingerprint and iris etc. are also used to provide security to the network users. Challenge-response scheme [CHALL] is a family of protocols in which one party presents a question ('challenge') and another party must provide a valid answer ('response') to be authenticated.

Password-Based Authentication Protocol

Password system [ADAM99] is the oldest and the most popular authentication scheme used in the modern world. Authentication systems work with password systems to make sure the users are who they say they are. Depending on the kind of password system used in authentication systems, the password files containing the master list of all passwords on an intranet can be plain text or encrypted.

Most attacks occur because an unauthorized person has managed to discover an authorized person's user name and password. However, making it hard for users to use passwords iscounterproductive and leads to increased vulnerability. The passwords of systems administrators or super users require special care, since if these passwords were compromised, the intruder would have full access to an intranet and all its corporate riches.

New servers often come with standard default passwords. However, it is really the fault of the systems administrators who fail to change the defaults. Similarly, care must be taken when, due to necessary technical work being done, technicians require root access or load custom utilities. Sometimes the default passwords are changed, and you think you are safe, but at some point during a disaster recovery process old users and / or passwords are loaded back in place.

Most systems require that passwords be changed periodically so that even if passwords are discovered or given out, there is only a limited window of vulnerability. People, of course, might try to circumvent this by changing their password and then changing it right back again. However, this can be prevented by systems requiring that when users change their passwords they must choose a password that they have not used before.

The logical extension of this "never before used" password requirement is the single -use

password. There are several methods of generating these passwords, including software and hardware methods. The software method still requires a truly secret password but it is used to generate a number of one-time variations that are used without encryption. The software method is still fundamentally a "something you know" type of protection. Hardware solutions add a "something you have" component, a physical device that generates single use passwords.

Biometric-Based Authentication Protocol

The growth and development [SARA11] of the Internet in the recent years has been very significant. But, the security and authentication is still a challenging problem. The security and authentication of the users in the Wireless LANs is also a serious issue. Hence the security of the network users has become a vital factor. There are various techniques available in the literature which make use of passwords, smart cards etc., to provide network related security. But these conventional authentication systems have lot of limitations. Most recently biometric features like fingerprint and iris are also used to provide security to the network users. These biometric features are very reliable compared to the traditional methods. Biometrics refers to the automatic identification of a person based on his/her physiological or behavioral characteristics. This method of identification is preferred over traditional methods involving passwords and PIN numbers for various reasons:

- The person to be identified is required to be physically present at the point-of-identification.
- Identification based on biometric techniques obviates the need to remember a password or carry a token.

The Biometric based user authentication [RAJE08] systems are very much secured and efficient to use and place total trust on the authentication server where biometric verification data are stored in a central database. This biometrics based user authentication system improves the network security. Some of the most extensively used biometric are hand geometry, face, fingerprint, retina, iris, DNA, signature and voice.

Biometrics is the knowledge [GUNA06] of measuring and statistically analyzing Biological data can be used to recognize different body parts like the eyes, fingerprints, facial characteristics, voice, iris etc. Thus, it takes security to the next level by not just confining it to authenticating passwords, iris matching techniques.

A conventional biometric authentication [YANS11] system consists of two phases: enrollment and verification is represented in Figure 1.6. During the enrollment phase, a biometric feature set is extracted from user's biometric data and a template is created and stored.

Challenge-Response Authentication Protocol

Challenge-Response Authentication [W8] is a method for proving your identity over an insecure medium without giving any information out to eavesdroppers that may enable them to identify themselves as you. It uses a cryptographic protocol [W9] that allows proving that the user knows the password without revealing the password itself. Using this method, the application first obtains a random challenge from the server. It then computes the response by applying a cryptographic hash function to the server challenge combined with the user's password. Finally, the application sends the response along with the original challenge back to the server. Because of the "one-way" properties of the hash function, it is impossible to recover the password from the response sent by the application.

Upon receiving the response, the server applies the same hash function to the challenge combined with its own copy of the user's password. If the resulting value matches the response sent by the application, this indicates with a very high degree of probability that the user has submitted the correct password.

Mutual authentication [W10] is performed using a challenge-response handshake in both directions; the server ensures that the client knows the secret, and the client also ensures that the server knows the secret, which protects against a rogue server impersonating the real server.

Challenge-response authentication can help solve the problem of exchanging session keys from encryption. Using a key derivation function, the challenge value and the secret may be combined to generate an unpredictable encryption key for the session.